USN : 1BM22CS259

LAB-5 : Grey Wolf Optimizer (GWO):

CODE:

```
import numpy as np
import matplotlib.pyplot as plt

# Step 1: Define the Problem (a mathematical function to optimize)
def objective_function(x):
    return np.sum(x**2)  # Example: Sphere function (minimize sum of squares)

# Step 2: Initialize Parameters
num_wolves = 5  # Number of wolves in the pack
num_dimensions = 2  # Number of dimensions (for the optimization problem)
num_iterations = 30  # Number of iterations
lb = -10  # Lower bound of search space
ub = 10  # Upper bound of search space

# Step 3: Initialize Population (Generate initial positions randomly)
wolves = np.random.uniform(lb, ub, (num_wolves, num_dimensions))

# Initialize alpha, beta, delta wolves
alpha_pos = np.zeros(num_dimensions)
beta_pos = np.zeros(num_dimensions)
delta_pos = np.zeros(num_dimensions)

alpha_score = float('inf')  # Best (alpha) score
beta_score = float('inf')   # Second best (beta) score
delta_score = float('inf')  # Third best (delta) score

# To store the alpha score over iterations for graphing
```

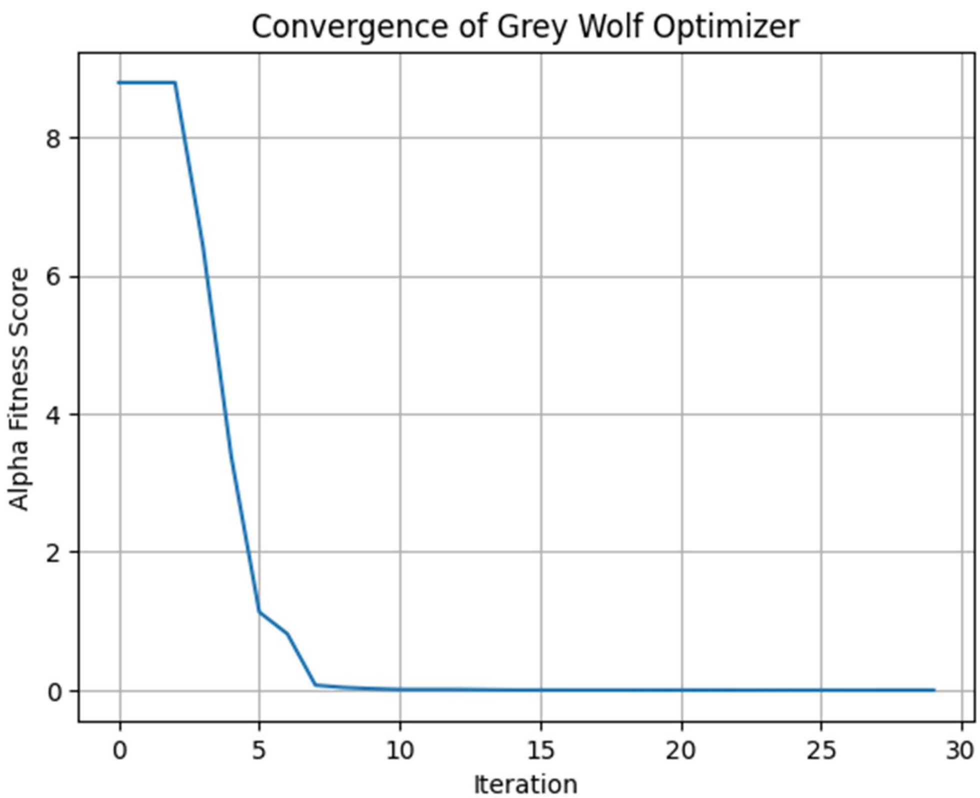
```
alpha_score_history = []

# Step 4: Evaluate Fitness and assign Alpha, Beta, Delta wolves
def evaluate_fitness():
    global alpha_pos, beta_pos, delta_pos, alpha_score, beta_score, delta_score

    for wolf in wolves:
        fitness = objective_function(wolf)

        # Update Alpha, Beta, Delta wolves based on fitness
        if fitness < alpha_score:
            delta_score = beta_score
            delta_pos = beta_pos.copy()

            beta_score = alpha_score
            beta_pos = alpha_pos.copy()

            alpha_score = fitness
            alpha_pos = wolf.copy()
        elif fitness < beta_score:
            delta_score = beta_score
            delta_pos = beta_pos.copy()

            beta_score = fitness
            beta_pos = wolf.copy()
        elif fitness < delta_score:
            delta_score = fitness
            delta_pos = wolf.copy()

# Step 5: Update Positions
def update_positions(iteration):
```

```
a = 2 - iteration * (2 / num_iterations)  # a decreases linearly from 2 to 0

for i in range(num_wolves):
    for j in range(num_dimensions):
        r1 = np.random.random()
        r2 = np.random.random()

        # Position update based on alpha
        A1 = 2 * a * r1 - a
        C1 = 2 * r2
        D_alpha = abs(C1 * alpha_pos[j] - wolves[i, j])
        X1 = alpha_pos[j] - A1 * D_alpha

        # Position update based on beta
        r1 = np.random.random()
        r2 = np.random.random()
        A2 = 2 * a * r1 - a
        C2 = 2 * r2
        D_beta = abs(C2 * beta_pos[j] - wolves[i, j])
        X2 = beta_pos[j] - A2 * D_beta

        # Position update based on delta
        r1 = np.random.random()
        r2 = np.random.random()
        A3 = 2 * a * r1 - a
        C3 = 2 * r2
        D_delta = abs(C3 * delta_pos[j] - wolves[i, j])
        X3 = delta_pos[j] - A3 * D_delta

        # Update wolf position
        wolves[i, j] = (X1 + X2 + X3) / 3
```

```
            # Apply boundary constraints
            wolves[i, j] = np.clip(wolves[i, j], lb, ub)

# Step 6: Iterate (repeat evaluation and position updating)
for iteration in range(num_iterations):
    evaluate_fitness()  # Evaluate fitness of each wolf
    update_positions(iteration)  # Update positions based on alpha, beta, delta

    # Record the alpha score for this iteration
    alpha_score_history.append(alpha_score)

    # Optional: Print current best score
    print(f"Iteration {iteration+1}/{num_iterations}, Alpha Score: {alpha_score}")

# Step 7: Output the Best Solution
print("Best Solution:", alpha_pos)
print("Best Solution Fitness:", alpha_score)

# Plotting the convergence graph
plt.plot(alpha_score_history)
plt.title('Convergence of Grey Wolf Optimizer')
plt.xlabel('Iteration')
plt.ylabel('Alpha Fitness Score')
plt.grid(True)
plt.show()
```

OUTPUT:

```
Iteration 1/30, Alpha Score: 8.789922247101906
Iteration 2/30, Alpha Score: 8.789922247101906
Iteration 3/30, Alpha Score: 8.789922247101906
Iteration 4/30, Alpha Score: 6.409956649485766
Iteration 5/30, Alpha Score: 3.383929841190778
Iteration 6/30, Alpha Score: 1.1292299489236237
Iteration 7/30, Alpha Score: 0.8136628488047792
Iteration 8/30, Alpha Score: 0.07110881373527288
Iteration 9/30, Alpha Score: 0.03823180120070083
Iteration 10/30, Alpha Score: 0.021111314445105462
Iteration 11/30, Alpha Score: 0.00874782100259989
Iteration 12/30, Alpha Score: 0.00874782100259989
Iteration 13/30, Alpha Score: 0.00874782100259989
Iteration 14/30, Alpha Score: 0.005066807028932165
Iteration 15/30, Alpha Score: 0.0011746187200998674
Iteration 16/30, Alpha Score: 0.0011746187200998674
Iteration 17/30, Alpha Score: 0.0008078646351838173
Iteration 18/30, Alpha Score: 0.0008078646351838173
Iteration 19/30, Alpha Score: 0.0006302256737926024
Iteration 20/30, Alpha Score: 0.0005272190797352655
Iteration 21/30, Alpha Score: 0.00035614966782860404
Iteration 22/30, Alpha Score: 0.0003270119398391142
Iteration 23/30, Alpha Score: 0.00022723766847392013
Iteration 24/30, Alpha Score: 0.00022152382849585967
Iteration 25/30, Alpha Score: 0.00022152382849585967
Iteration 26/30, Alpha Score: 0.00020102313789207912
Iteration 27/30, Alpha Score: 0.0001974565833678501
Iteration 28/30, Alpha Score: 0.0001547675581999543
Iteration 29/30, Alpha Score: 0.00014751518222697009
Iteration 30/30, Alpha Score: 0.00014751518222697009
Best Solution: [ 0.00643925 -0.01029812]
Best Solution Fitness: 0.00014751518222697009
```



Convergence of Grey Wolf Optimizer
Alpha Fitness Score
Iteration
0
2
4
6
8
0
5
10
15
20
25
30